Package ‘multiplex’

August 16, 2017

Type Package
Version 2.7
Depends R (>= 3.3.1)
Imports methods
Suggests multigraph, Rgraphviz, knitr
Title Algebraic Tools for the Analysis of Multiple Social Networks
Description Algebraic procedures for the analysis of multiple social networks are delivered with this package. Among other things, it makes possible to create and manipulate multivariate network data with different formats, and there are effective ways available to treat multiple networks with routines that combine algebraic systems like the partially ordered semigroup or the semiring structure together with the relational bundles occurring in different types of multivariate network data sets. It also provides an algebraic approach for two-mode networks through Galois derivations between families of the pairs of subsets in the two domains.
Date 2017-08-16
Author Antonio Rivero Ostoic [aut, cre]
Maintainer Antonio Rivero Ostoic <multiplex@post.com>
URL http://github.com/mplex/multiplex/
BugReports https://github.com/mplex/multiplex/issues/
Repository CRAN
Encoding latin1
License GPL-3
VignetteBuilder knitr
NeedsCompilation no
Date/Publication 2017-08-16 09:53:37 UTC

R topics documented:

 multiplex-package .................................................. 3
 as.semigroup .......................................................... 5
<table>
<thead>
<tr>
<th>R topics documented:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>as.signed</td>
<td>6</td>
</tr>
<tr>
<td>as.strings</td>
<td>7</td>
</tr>
<tr>
<td>bundle.census</td>
<td>8</td>
</tr>
<tr>
<td>bundles</td>
<td>9</td>
</tr>
<tr>
<td>cngr</td>
<td>11</td>
</tr>
<tr>
<td>comps</td>
<td>12</td>
</tr>
<tr>
<td>convert</td>
<td>13</td>
</tr>
<tr>
<td>cph</td>
<td>14</td>
</tr>
<tr>
<td>decomp</td>
<td>15</td>
</tr>
<tr>
<td>diagram</td>
<td>17</td>
</tr>
<tr>
<td>diagram.levels</td>
<td>18</td>
</tr>
<tr>
<td>dichot</td>
<td>19</td>
</tr>
<tr>
<td>edgeT</td>
<td>20</td>
</tr>
<tr>
<td>expos</td>
<td>21</td>
</tr>
<tr>
<td>fltr</td>
<td>23</td>
</tr>
<tr>
<td>galois</td>
<td>24</td>
</tr>
<tr>
<td>hierar</td>
<td>25</td>
</tr>
<tr>
<td>iinc</td>
<td>26</td>
</tr>
<tr>
<td>incubs</td>
<td>27</td>
</tr>
<tr>
<td>is.mc</td>
<td>28</td>
</tr>
<tr>
<td>isom</td>
<td>29</td>
</tr>
<tr>
<td>ltw</td>
<td>30</td>
</tr>
<tr>
<td>mnpxl</td>
<td>31</td>
</tr>
<tr>
<td>neighb</td>
<td>32</td>
</tr>
<tr>
<td>pacnet</td>
<td>33</td>
</tr>
<tr>
<td>partial.order</td>
<td>35</td>
</tr>
<tr>
<td>perm</td>
<td>36</td>
</tr>
<tr>
<td>pi.rels</td>
<td>37</td>
</tr>
<tr>
<td>prev</td>
<td>38</td>
</tr>
<tr>
<td>rbox</td>
<td>39</td>
</tr>
<tr>
<td>read.dl</td>
<td>40</td>
</tr>
<tr>
<td>read.gml</td>
<td>41</td>
</tr>
<tr>
<td>read.srt</td>
<td>42</td>
</tr>
<tr>
<td>reduc</td>
<td>44</td>
</tr>
<tr>
<td>rel.sys</td>
<td>45</td>
</tr>
<tr>
<td>relabel</td>
<td>47</td>
</tr>
<tr>
<td>rm.isol</td>
<td>48</td>
</tr>
<tr>
<td>semigroup</td>
<td>49</td>
</tr>
<tr>
<td>semiring</td>
<td>51</td>
</tr>
<tr>
<td>signed</td>
<td>52</td>
</tr>
<tr>
<td>strings</td>
<td>54</td>
</tr>
<tr>
<td>summaryBundles</td>
<td>55</td>
</tr>
<tr>
<td>transf</td>
<td>56</td>
</tr>
<tr>
<td>wordT</td>
<td>58</td>
</tr>
<tr>
<td>write.dat</td>
<td>59</td>
</tr>
<tr>
<td>write.dl</td>
<td>60</td>
</tr>
<tr>
<td>write.gml</td>
<td>61</td>
</tr>
<tr>
<td>write.srt</td>
<td>62</td>
</tr>
</tbody>
</table>
Description

One of the aims of the ‘multiplex’ package is to meet the necessity to count with an analytic tool specially designed for social networks with relations at different levels. In this sense, ‘multiplex’ counts with functions that models the local role algebras of the network based on the simple and compound relations existing in the system, and also a procedure for the construction and analysis of signed networks through the semiring structure. The different relational patterns at the dyadic level in the network can be obtained as well, which can serve for a further analysis with different types of structural theories.

It is also possible to take the attributes of the actors in the analysis of multiple networks with different forms to incorporate this kind of information to the existing relational structures. In this case for example the network exposure of the actors can be taken in the context of multiple networks, or else the attributes can be embedded in the resulted algebraic structures.

Details

Package: multiplex
Type: Package
Version: 2.7
Date: 16 August 2017
License: GPL-3
LazyLoad: yes

To work with this package we typically start with a specific algebraic structure. A semigroup is a closed system made of a set of elements and an associative operation on it. This algebraic structure is constructed by the \texttt{semigroup} function, and it takes an array of (usually but not necessarily) multiple binary relations, which are the generator relations. The \texttt{Word Table} and the \texttt{Edge Table} serve to describe completely the semigroup, and they are constructed with the functions \texttt{wordt} and \texttt{edget} respectively. Unique relations of the complete semigroup are given by the \texttt{strings} function. The \texttt{partialNorder} function specifies the ordering of the string elements in the semigroup. The \texttt{function diagram} produces the lattice of inclusions of a structure having ordered relations.

Semigroups can be analysed further by \texttt{ltlw} function, and they also can be reduced by a decomposition process. The decomposition is based on congruence or $\pi$-relations of the unique strings imported from Pacnet. In this case \texttt{pi.rels}, \texttt{cngr}, and \texttt{decomp} will make this job for you either for an abstract or a partially ordered structure.

In addition, it is possible to analyse structural balance in signed networks, which are built by \texttt{signed}, through the algebraic structure of the semiring. A semiring is an algebraic structure that combines an abstract semigroup with identity under multiplication and a commutative monoid un-
The semiring function is capable to perform both balance and cluster semiring either with cycles or semicycles.

There are other capabilities in the package that are not strictly algebraic. For instance, the dichtot serves to dichotomize the input data with a specified cut-off value, rm.isol removes isolated nodes, and the perm function performs an automorphism of the elements in the representative array. All these functions are built for multiple networks represented by high dimensional structures that can be constructed by the function zbind.

The ‘multiplex’ package creates a Relation-Box with the rbox function, and it implements the Partial Structural Equivalence expressed in the cumulated person hierarchy of the network calculated via the cph function.

Relational bundles are identified through the bundles function, which provides lists of pair relations. The transf function serves to transform such data into a matrix form. The enumeration of the different bundle classes is given by bundle.census. An advantage of counting with the bundle patterns is that the different types of bundles serve to establish a system inside the network, in which it is possible to measure the network exposure in multivariate relational systems. Such features can be realized via the rel.sys and expos functions respectively. Several attributes can be derived by galois, which provides an algebraic approach for two-mode networks.

Finally, multivariate network data can be created through the (s)end (r)eve (t)ies format that can be loaded and transformed via the read.srt function. Other formats for multiple network data like Ucinet dl or Visone gml can be imported and exported as well with the ‘multiplex’ package.

Author(s)
J. Antonio Rivero Ostoic
Maintainer: Antonio Rivero Ostoic <multiplex@post.com>

References
Lorrain, François and Harrison C. White, ‘Structural Equivalence of Individuals in Social Networks.’ *Journal of Mathematical Sociology*, 1, 49-80. 1971

See Also
multigraph, bmgraph

Examples
```r
## Create the data: two binary relations among three elements
arr <- round( replace( array(runif(18), c(3,3,2)), array(runif(18), c(3,3,2))>.5, 3 ) )
```
## as.semigroup

```r
## Dichotomize it with customized cutoff value
dichot(arr, c = 3)

## preview
prev(arr)

## create the semigroup (elay...)
semigroup(arr)

## and look at the strings
strings(arr)
```

### Description

A generic function for coercing an R object to a `semigroup` class.

### Usage

```r
as.semigroup(x, lbs)
```

### Arguments

- `x` an array representing the semigroup
- `lbs` (optional) label strings for the semigroup

### Details

Since many of the functions in the `multiplex` package require an object of the 'Semigroup' class, this function produces this class object from an array representing the semigroup structure.

### Value

An object of the 'Semigroup' class

- `ord` a number with the dimension of the semigroup
- `st` the strings, i.e. a vector of the unique relations
- `S` the semigroup of relations

### Note

In this function the input object typically comes from the `convert` function.
as.signed

Author(s)
Antonio Rivero Ostoic

See Also
convert, semigroup

Examples

```r
## scan the multiplication table data
s <- matrix(c(1, 1, 1, 3, 3, 3, 3, 3), nrow=3, ncol=3, byrow=TRUE)

## make a semigroup object
as.semigroup(s)
```

as.signed

Coerce to aSigned Object

Description
A generic function for coercing an object to a Signed class.

Usage

```r
as.signed(x, lbs)
```

Arguments

- `x`: a matrix representing the signed network
- `lbs`: (optional) labels for the signed matrix

Details
Since the semiring function requires an object with a `Signed` class, this function produces this class object from an array representing the signed network

Value
The array as a Signed class

See Also
signed, semiring
Examples

```r
## Load the data
data("

## Coerce parts of the signed matrix with two types of relations
as.signed(signed(incubA$IM)$s[1:2,1:2])
```

---

**as.strings**  
*Coerce an Object to a Strings Class*

**Description**

A generic function for coercing an R object to a `Strings` class.

**Usage**

```r
as.strings(x, lbs = NULL)
```

**Arguments**

- `x`: an array; usually with three dimensions of stacked matrices where the multiple relations are placed.
- `lbs`: (optional) the labels of the strings

**Details**

This function is useful to proceed with the establishment of the partial order in the strings of relations where the object should be of a `Strings` class.

**Value**

An object of `Strings` class

- `wt`: the word tables
- `ord`: the number of unique relations in the semigroup

**Author(s)**

Antonio Rivero Ostoic

**See Also**

`strings`, `partial.order`, `zbind`
Examples

```r
## Create the data: two sets with a pair of binary relations among
## three elements
arr1 <- round(replace(array(runif(18), c(3,3,2)), array(runif(18),
c(3,3,2))>.5, 3 ))

arr2 <- round(replace(array(runif(18), c(3,3,2)), array(runif(18),
c(3,3,2))>.5, 3 ))

## bind the data sets
arrs <- zbind(arr1, arr2)

## make the data a strings object
as.strings(arrs)
```

<table>
<thead>
<tr>
<th>bundle.census</th>
<th>Bundle Census</th>
</tr>
</thead>
</table>

Description

A function to perform the Bundle Census in a given multiple network.

Usage

`bundle.census(x, loops = FALSE)`

Arguments

- **x**: an array; usually with three dimensions of stacked matrices where the multiple relations are placed.
- **loops**: (logical) whether or not the loops should be considered

Details

This function calculates the number of occurrences for each bundle class pattern in the multiple network. A bundle is a special type of pattern made of relations at different levels that is binding a pair of nodes or actors. Depending on the direction and occurrence of each possible tie, then it is possible to count with seven dyadic configuration classes, which are included in the census.

Value

A table with the occurrences in the distinct bundle class patterns. The first column in the output gives the number of bundles in the network excluding the null pattern, and then the totals for each bundle class pattern are specified in the following columns. If loops were considered, these are at the end of the table.

More detailed information about the bundle class occurrences is given by the function `bundles`. 
Note

Neither loops or null dyads are regarded as properly bundle classes.

Author(s)

Antonio Rivero Ostoic

References


See Also

bundles, summaryBundles

Examples

```r
## Create the data: two binary relations among three elements
arr <- round(replace(array(runif(18), c(3,3,2)), array(runif(18), c(3,3,2)), .8, 3 ))

## Calculate the Bundle Census
bundle.census(arr)
```

---

**bundles**

**Bundle Class Patterns**

**Description**

Classify the Bundle class patterns in a system of multiple relations

**Usage**

```r
bundles(x, loops = FALSE, smpl = FALSE, lb2lb = TRUE,
        collapse = FALSE, sep)
```

**Arguments**

- `x` : an array; usually with three dimensions of stacked matrices where the multiple relations are placed.
- `loops` : (logical) whether or not the loops should be considered as a particular bundle
- `smpl` : (logical) simplify the strings of relations? Default no.
- `lb2lb` : (logical) should the labels of the nodes be included in the output? Default yes.
- `collapse` : (logical) collapse the distinct levels of relations in the network? Default no.
- `sep` : (optional) the pair separator for the pairwise relations
Details

A bundle is a special type of pattern made of relations at different levels that is binding a pair of nodes or actors in a network of relations. A bundle class is a dyadic configuration resulting from the mixture of the direction and the types of tie between the nodes or actors. There are in total seven dyadic configuration classes, which are null, asymmetric, reciprocal, tie entrainment, tie exchange, mixed, and the full bundle pattern. This function provides the detailed information about the bundle class patterns in the multiple network as lists of pair relations among the nodes or actors, except for the ‘null’ pattern. In case that the nodes are not labeled, then an identification number will be assigned according to the nodes’ location in the array representation and as well when the lb2lb option is set to FALSE. This function assumes that the network is directed, and self ties can also be considered in the output. Long string labels can be simplified with smpl, whereas the collapse option blurs the levels in the strings. The input array is always dichotomized.

Value

An object of ‘Rel.Bundles’ class with the distinct bundle class patterns.

- **asym**: asymmetric
- **recp**: reciprocal
- **tent**: tie entrainment
- **txch**: tie exchange
- **mixed**: mixed
- **full**: full
- **loops**: loops (if chosen)

Note

It is also possible to obtain the total number of occurrences in each bundle class pattern by the `bundle.census` function.

Author(s)

Antonio Rivero Ostoic

References


See Also

`bundle.census, summaryBundles, transf.`
Examples

```r
## Create the data: two binary relations among three elements
arr <- round(replace(array(runif(18), c(3,3,2)), array(runif(18), c(3,3,2)) > .8, 3 ))

## Establish the different bundles
bundles(arr)
```

---

### cngr

#### Congruence Relations

**Description**

Find the congruence relations of a given abstract or a partially ordered semigroup.

**Usage**

```r
cngr(S, PO = NULL, unique = FALSE)
```

**Arguments**

- `S` : an object from the ‘Semigroup’ class.
- `PO` : (optional) the partial order table
- `unique` : (logical) whether or not return the unique congruence relations

**Details**

Congruences are equivalence relations that preserve the operation between the correspondent classes in the algebraic structure. In this case the different congruence classes are based on the substitution property of the semigroup object.

**Value**

An object of ‘Congruence’ class. The items included are:

- `S`: semigroup of relations
- `PO`: partial order table (if specified)
- `clu`: congruence classes

**Note**

If the partial order is supplied in the input, then the computation of the congruence classes is slightly faster than for an abstract semigroup.

**Author(s)**

Antonio Rivero Ostoic
References


See Also
decomp, pacnet

Examples

```r
## Create the semigroup object
arr <- round(replace(array(runif(18), c(3,3,2)), array(runif(18), c(3,3,2))>.5, 1))
#
S <- semigroup(arr)

## look at the congruences in S
cngr(S, P0=NULL)
```

---

**comps**  
*Find components in multiple networks*

**Description**

Function to find different components in the network plus isolates

**Usage**

```r
comps(x, bonds = c("entire", "strong", "weak"))
```

**Arguments**

- `x`: array representing the network
- `bonds`: the type of bonds to be used in the creation of the relational system for the different components

**Details**

The network's different components are obtained by means of transitive closure of the bundle ties. By default the 'entire' network is chosen, but the option `bonds` allows to discriminate different types of relational bundles.

**Value**

A list with two possible “components”

- `com`: a component
- `isol`: the isolates
convert

Author(s)

Antonio Rivero Ostoic

See Also

bundles, rel.sys

Examples

```r
## Create the data: two binary relations among three elements
arr <- round(replace(array(runif(18), c(3, 2)), array(runif(18),
                        c(3, 2)) > .9, 3))

## Find components ans isolates
comps(arr)
```

---

**convert**

*Convert Format of a Semigroup Object*

**Description**

A function to convert an object representing a semigroup of relations from Numerical to Symbolic format and vice versa.

**Usage**

```r
convert(x, lbs = NULL, SemigroupClass = FALSE)
```

**Arguments**

- `x` an object from the `Semigroup` class.
- `lbs` (optional) the labels to be used in the conversion
- `SemigroupClass` (logical) whether the output should be a `Semigroup` class

**Details**

In order to avoid the construction of the semigroup from primitive relations (since it might take a relative long time), this function is aimed to convert the semigroup structure in a fast way from a numerical to a symbolic form or vice versa.

**Value**

An array representing the semigroup of relations or an object of the `Semigroup` class.
Note

In case that the input data is in ‘symbolic’ form, then labels is ignored, and a ‘Warning’ message is printed in the output.

Author(s)

Antonio Rivero Ostoic

See Also

semigroup, as.semigroup, relabel

Examples

```r
## Create the multiplication table
s <- matrix(data=c(1, 1, 1, 3, 3, 3, 3, 3), nrow=3, ncol=3, byrow=TRUE)

## make a semigroup object with labels
S <- as.semigroup(s, lbs=c('n', 'm', 'n'))

## convert the semigroup to a symbolic form
convert(S)
```

Description

A function to calculate the Cumulated Person Hierarchy in networks of multiple relations

Usage

cph(W, lbs)

Arguments

- `W` an object of the ‘Rel.Box’ class.
- `lbs` (optional) the labels of the relational system

Details

The cumulated person hierarchy is used to determine the partial structural equivalence among the actors in a multiple network. Two nodes are considered as *partial structural equivalent* iff they have identical role sets.

The outcome of this function depends on the characteristics of the Relation-Box.
decomp

Value

An object of ‘Partial.Order’ class with an array representing the cumulated person hierarchy.

Note

If the length of the labels differ from the order of the relational system, then labels will be ignored.

Author(s)

Antonio Rivero Ostoic

References


See Also

rbox, semigroup, diagram

Examples

## Create the data: two binary relations among three elements
arr <- round( replace( array(runif(18), c(3,3,2)), array(runif(18), c(3,3,2))>.9, 3 ) )

## Make the relation box
rarr <- rbox(arr)

## Calculate the cumulated person hierarchy
cph(rarr)

decomp  Decomposition of a Semigroup Structure

Description

A function to perform the decomposition of a semigroup structure

Usage

decom(S, x, type = c("mc", "pi", "cc"), reduc = FALSE)
Arguments

- **S**: an object of a ‘Semigroup’ class
- **x**: either an object of a ‘Congruence’ class or an object of a ‘Pi.rels’ class
- **type**: whether the reduction is based on a congruence class or rather on a $\pi$-relation or a meet-complement in the ‘Pi.rels’ class
- **reduc**: (logical) does the return object should include the reduced structures?

Details

The `decomp` function is a reduction form of an algebraic structure like the semigroup that verifies which of the class members in the system are congruent to each other. The decomposed object then is made of congruent elements, which form part of the lattice of congruence classes in the algebraic structure. In case that the input data comes from the Pacnet program, then such elements are in form of $\pi$-relations or the meet-complements of the atoms; otherwise these are simply equivalent elements satisfying the substitution property.

Value

An object of ‘Decomp’ class having:

- **clu**: vector with the class membership
- **eq**: the equations in the decomposition
- **IM**: (optional) the image matrices
- **PO**: (optional) the partial order table
- **dims**: (optional) a vector with the dimensions of the image matrices

Note

Reduction of the partial order table should be made by the `reduc` function.

Author(s)

Antonio Rivero Ostoic

References


See Also

`cngr`, `reduc`, `pi.rels`, `semigroup`, `partial.order`
plot hasse diagram of a set of ordered relations

description

A function to plot the Hasse Diagram of partially ordered relations.

usage

diagram(x, unord = TRUE, attrs = NULL, main = NULL, cex.main = graphics::par()
bg = graphics::par()

arguments

x a matrix representing ordered relations
unord (logical) whether or not the elements not belonging in the order should be included in the diagram
attrs (optional) attributes of the diagram
main (optional) title of the diagram
cex.main (optional) size of the diagram’s title
bg (optional) the background color of the diagram

Details

An example of ordered relations is found in the partial order table of relations, which is product of the 'strings' option in the partial.order function. Another set of ordered relations comes from the table produced on Galois derivations in the mentioned function.

In either case this function plot either the partial order or a linear order diagram, depending on the results as Hasse diagrams.

Value

A Hasse diagram of the partial order relation.

Warning

This function requires that the Rgraphviz package is available.

Note

Note that if the elements of the partial order are not labelled, Roman numerals will be given to each element.
Author(s)
Antonio Rivero Ostoic

See Also
partial.order, as.strings, strings, diagram.levels, galois.

Examples

```r
## load the data
data("incubA")

## given e.g. a partial order table in the object 'po'
po <- partial.order(as.strings(incubA$IM), type="strings")

## plot the order relation as a Hasse diagram.
## Not run: if(require(Rgraphviz)) {
## plot(diagram(po))
## }
## End(Not run)
```

---

diagram.levels Levels in the Lattice Diagram

Description
This is a function that reads the different levels in the lattice diagram of the partial order structure among actors and ties in the network

Usage
diagram.levels(x, perm = FALSE)

Arguments

- `x` A matrix representing the partial order
- `perm` (optional) whether or not to return the permuted structure

Details
When it comes to reduce the structure of a multiple network, many times the partial order structure provides different classes of elements depending in the inclusions these elements have. In this sense, the illustration given by the diagram function provides us typically with different levels of the ordered relations, which are read by this routine.
**dichot**

**Value**

A data frame with the elements of the partial order structure with the column names indicating the element class. If the permutation is specified, then a vector with the levels and a matrix with the permuted structure are given as well.

**Note**

This function requires that the `Rgraphviz` package is available. Besides, since the `pictex` function from `grDevices` is inside this routine, it implies counting with administrator privileges for running.

**Author(s)**

Antonio Rivero Ostoic

**See Also**

`partial.order`, `diagram`, `perm`

**Examples**

```r
## load the data
data("incubA")

## given e.g. a partial order table in the object 'po'
po <- partial.order(as.strings(incubA$IM))

## find the levels in the lattice diagram
## Not run: diagram.levels(po)
```

---

### dichot

**Dichotomize data with a cutoff**

**Description**

Function to dichotomize the input data for the semigroup construction with a cutoff value.

**Usage**

```r
dichot(x, c = 1, diag.incl = TRUE)
```

**Arguments**

- `x`: some data in a numeric form (usually arrays)
- `c`: the cutoff value to perform the dichotomization (default 1)
- `diag.incl`: (logical) whether or not the diagonals should be included (default TRUE)
**Details**

This is a convenient function (or wrapper if you like) of the `replace` function. In this case the function is aimed to specify a cutoff value for the dichotomization of the data where the values equal or higher to the cutoff are converted to one, while the others are set to zero. The cutoff can be any real number.

**Value**

Binary values of the input data.

**Note**

The labels are preserved after the dichotomization.

**Author(s)**

Antonio Rivero Ostoic

**See Also**

`replace, prev, semigroup`.

**Examples**

```
## Create the data: 2 binary relations among 3 elements
arr <- round(replace( array(runif(18), c(3,3,2)), array(runif(18), c(3,3,2))>.5, 3 ) )

## dichotomize it with a cutoff value of 2
dichot(arr, c = 2)
```

---

**edgeT**  
*Edge Table Generator*

**Description**

The Edge Table generator of multiple relations.

**Usage**

`edgeT(x)`

**Arguments**

- `x` an array; usually with three dimensions of stacked matrices where the multiple relations are placed.
Details

The Edge Table is the complete right multiplication table of the semigroup having its elements for each of its generators.

Value

gens the generator relations
ET the Edge Table

Author(s)

Antonio Rivero Ostoic

References


See Also

wordT, semigroup.

Examples

```r
## Create the data: 2 binary relations among 3 elements
class <- round(replace(array(runif(18), c(3,3,2)), array(runif(18),
c(3,3,2))>.5, 1 ))

## get the edge table
edget(class)
```

expos 21

Description

Function to measure the network exposure of the nodes according to a chosen relational system representing the multiple network.

Usage

```r
expos(rs, classes = FALSE, allClasses = FALSE, allNodes = TRUE)
```
Arguments

rs an object of ‘Rel.System’, typically with node attributes.

classes (optional) whether or not should be included in the output the categories of adopters

allClasses (optional) whether or not to include empty classes within the categories of adopters. Ignored if classes is FALSE

allNodes (optional) whether or not to include all actors in the network regardless they are in the chosen system. Ignored if classes is FALSE

Details

This is a generalization of the network exposure measure for multiple networks with the characteristics chosen for the representative relational system. Such system can be the entire network or configuration with strong or weak bonds among the actors. It is possible to specify different behaviours of the nodes representing social actors, which are indicated in the form of a relational system. The network exposure measure is calculated according to the immediate neighbours to the reference actor.

Value

Classes if classes is set to TRUE, the adoption membership for the type of relational system chosen, including isolated actors in the system.

Bonds the type of bonds of the relational system (cf. rel.sys)

Exposure the exposure to the attribute(s) for acquisition through immediate neighbour relations

Author(s)

Antonio Rivero Ostoic

References


See Also

rel.sys, neighb, bundles

Examples

## Create the data: two binary relations among three elements
arr <- round( replace( array( runif(18), c(3,3, 2) ), array( runif(18), c(3, 3, 2) ) > .9, 3 ) )
fltr

## the first array is for attributes
rs <- rel.sys(arr, att = 1)

## Calculate the exposure measure for an attribute type with adopter categories
expos(rs, classes = TRUE)

---

**fltr**  

**Principal filters**

### Description

A function to find principal filters in a partial order

### Usage

```r
fltr(x, PO, rclos = TRUE, ideal = FALSE)
```

### Arguments

- `x`: the reference element in the partial order (integer or character)
- `PO`: the partial order
- `rclos`: (logical) apply reflexive closure?
- `ideal`: (logical) whether or not the “filter” is an ideal

### Details

This function helps to find principal filters or principal ideals for an element in a partial order structure. Such inputs are typically a concept or an object or attribute in the concept together with the associated partial ordering structure of the concepts, which results from Galois derivations. Typically if the reference element refers to a concept, then it is given as a positive integer indicating the concept label. Another option is to refer an object or an attribute by a character name, which should be part in the labels of the dimensions of the partial order table with a reduced labeling. Principal filters with a full labelling are not allowed if the reference element is an object or an attributes. Use an integer for the concept instead.

### Value

A named list with the elements in the upset or downset of the principal filter or ideal corresponding to the reference element in the partial order.

### Author(s)

Antonio Rivero Ostoic

### References

galois

See Also
galois, partial.order, diagram.

Examples

```r
## Create a data frame
dfr <- data.frame(x=1:3, y=5:7)

## Partial ordering of concepts
PO <- partial.order(galois(dfr),"galois")

## Filter for the first element
fltr(1, PO, rclos=TRUE)
```

galois

Galois derivations between subsets

Description

Function to perform Galois derivations between partially ordered subsets

Usage

galois(x, labeling = c("full", "reduced"))

Arguments

- `x` a data frame with objects and attributes
- `labeling` whether the derivations should be with full or reduced labeling

Details

Galois derivations (or connections) are mappings between families of partially ordered subsets of elements. Such derivations are useful to analyse the structure of both subsets, which in a social network are typically the actors and their corresponding affiliations or events, i.e. two-mode networks, or else a group of objects with a list of different attributes used in formal concept analysis.

Value

A labelled list with Galois derivations of objects and attributes

Note

Full labeling implies first objects and then attributes, whereas the reduced option is given the other way around.
Description
A function to establish either the Person or the Relation Hierarchy in a multiple network

Usage
hierar(W, x, type = c("person", "relation"))

Arguments
W
an object of ‘Rel.Box’
x
(integer or character) the actor of reference, either by its location in the adjacency matrix or by the label.
type
whether the hierachy is for the ‘persons’ or for the ‘relations’ in the network with respecto to ‘x’

Details
The person hierarchy refers to the inclusion relations among the actors, whereas the relation hierarchy refers to the inclusion relations among the ties. Both are from the perspective of a chosen actor of reference in the given network.

Value
An array that represents the partial order structure of the respective hierarchy.


Note

The cumulation of the person hierarchy is obtained through the `cph` function.

Author(s)

Antonio Rivero Ostoic

References


See Also

`rbox, cph, partial.order, diagram`

Examples

```r
## Create the data: 2 binary relations among 3 elements
arr <- round( replace( array( runif(18), c(3, 3, 2) ), array( runif(18), c(3, 3, 2) ) > .5, 3 ) )

## The relation box
rarr <- rbox(arr, k=1)

## Calculated the person hierarchy of a random actor
hierar(rarr, ceiling(runif(1, min=0, max=dim(arr)[2])))
```

---

### `iinc` Induced Inclusions

**Description**

Function to establish class membership of induced inclusions given a partial order structure

**Usage**

`iinc(inc, PO, equat = FALSE, sep)`

**Arguments**

- `inc` : an induced inclusion with a 'list' format
- `PO` : matrix with the partial order structure
- `equat` : (logical) should the equations be printed in the outcome?
- `sep` : (optional) the pair separator for the pairwise relations
Details

The induced inclusion function takes certain additions to the partial order, and then check whether a congruence relation is established among the relations.

Value

A vector with the class membership of the elements in the structure

Author(s)

Antonio Rivero Ostoic

References


See Also

partial.order, semigroup, pacnet.

Examples

```r
## Load the data, and obtain the partial order
data("incubA")
po <- partial.order(strings(incubA$IM))

## Get a cluster vector given induced inclusion on the PO
iinc(transf(1-po), po)
```

### Description

These are four data sets collected in year 2010 (see ‘source’ for the details) of multiple relations between entrepreneurial firms working in business incubators in Denmark.

Each data set contains the adjacency matrices of the three social relations, coded as C, F, and K for working collaboration, informal friendship, and perceived competition among the firms. There are also two actor attributes corresponding to the adoption of a Web innovations at that time by the firms where A stands for Linkedin and B for Facebook.

In addition, there is a blockmodel attached to each data set that is product of Compositional Equivalence (cf. `cph`) with transposes for each type of social tie labelled with the following letter in the Latin alphabet; i.e. D for collaboration, etc.
Usage

data("incubs")
data("incubA")
data("incubB")
data("incubC")
data("incubD")

Format

Each data set is a list with a pair of three dimensional arrays.
For incuba, the dimensions of net are $26 \times 26 \times 5$, and of IM are $4 \times 4 \times 7$ (the two attributes led to the identity matrix).
For incubB, the dimensions of net are $18 \times 18 \times 5$, and of IM are $4 \times 4 \times 8$.
For incubC, the dimensions of net are $22 \times 22 \times 5$, and of IM are $3 \times 3 \times 8$.
For incubD, the dimensions of net are $15 \times 15 \times 5$, and of IM are $4 \times 4 \times 6$.
All four networks are putted in together in incubs.

Source


---

is.mc Are Two $\pi$-relations Meet Complements?

Description

Function to find whether two $\pi$-relations are Meet Complement of a given an Atom.

Usage

is.mc(B, C, A, ord = NULL)

Arguments

<table>
<thead>
<tr>
<th>B</th>
<th>one $\pi$-relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>another $\pi$-relation</td>
</tr>
<tr>
<td>A</td>
<td>the Atom</td>
</tr>
<tr>
<td>ord</td>
<td>(optional) the order of the semigroup</td>
</tr>
</tbody>
</table>

Details

Two congruent or $\pi$-relations are meet complement to a join irreducible element of the lattice of congruence relations whenever their greatest lower bound is the minimal element in the congruence lattice.
isom

Value
A logical indicating whether the input data are meet-complements of the Atom.

Note
The congruence relations must be in a 'list' format.

Author(s)
Antonio Rivero Ostoic

References

See Also
iinc, cngr, decomp, pi.rels

---

Description
Make isomorphic a vector or a data frame object.

Usage
```
isom(x, uniq = FALSE)
```

Arguments
- `x`: a vector or a data frame
- `uniq`: (logical) whether or not the unique class should be returned

Details
This function serves to normalize otherwise arbitrary numbering of a vector or a data frame object data into a progressive ordinal recount of the data in the specified object. This is especially useful to specify more properly the corresponding clustering vector of a set of elements in a given system.

Value
The input data with isomorphic classes.

In case that uniq is specified, then a list with the following items is given:

- `ism`: the input data with isomorphic classes
- `uniq`: the unique isomorphic classes
Author(s)

Antonio Rivero Ostoic

See Also

cngr, reduc

Examples

```r
## Create a data frame (from 'Vectors')
df <- data.frame(x=1:3, y=5:7)

## Make isomorphic the observations
isom(df)
```

First- and Last Letter Laws

Description

Find the First- and Last Letter Laws in the semigroup of relations.

Usage

```r
ltlw(x)
```

Arguments

- `x`: an object from the 'Semigroup' class.

Details

The First- and Last Letter Laws of the semigroup of relations correspond to the right and left zero of the semigroup, and it has some consequences in the interpretation of the results of the relational structure of the network. This function depends on the semigroup of a (recommended) labeled set of relationships.

Value

- `S`: the semigroup of relations.
- `strings`: the strings of relations in the semigroup.
- `First.Letter`: the First.Letters in rows and columns
- `Last.Letter`: the Last.Letters in rows and columns

Note

Since this function is based on the semigroup construction, see the Warning section and look at other details in the `semigroup` function section.
A function to transform multiple networks into a monoplex structure

Usage

\[
\text{mnplx}(\text{net}, \text{directed} = \text{TRUE}, \text{dichot} = \text{TRUE}, \text{diag.incl} = \text{FALSE})
\]

Arguments

- **net**: a three-dimensional array to be transformed into a matrix
- **directed**: (optional) whether to make the matrix symmetric or not
- **dichot**: (optional) should the output be dichotomized?
- **diag.incl**: (optional) should the diagonals be included?

Details

With this function it is possible to collapse multiple types of tie into a matrix representation with monoplex relations.
Value

A matrix of monoplex relations

Author(s)

Antonio Rivero Ostoic

See Also

zbind, dichot, reduc

Examples

```r
## Create the data: 2 binary relations among 3 elements
arr <- round(replace(array(runif(18), c(3,3,2)), array(runif(18), c(3,3,2))>.5, 1))

## Make it monoplex
mnplx(arr)
```

---

**neighb**

*Neighborhood of an actor or group of actors*

Description

A function to find the neighborhood of an actor or group of actors with a customized distance.

Usage

```r
neighb(x, rs, type = c("und", "inn", "out"), inclx = FALSE, k = 1, expand = FALSE)
```

Arguments

- `x` the reference actor labeled in `rs` or a vector of several actors
- `rs` the relational system of the network
- `type` whether the network is *undirected* (default) und; directed with *incoming node’s ties* inn to the the reference actor, or else with *outgoing arcs* out
- `inclx` (logical) should the reference actor be included in the output
- `k` the “distance” of the neighbor nodes to the reference actor (where k=1 gives the adjacent nodes)
- `expand` (logical) should the output be given by `k` (it only makes sense when k>1)
Details

The relational system serves to represent either the entire multiple network, or else just the relational bundles having a mutual or an asymmetric character. In this sense, this function detects the adjacent nodes to x according to the specified relational system, but as well the neighbours of the adjacent nodes with a customized length. Eventually, when the longest path or chain is reached, adding more value to k obviously will not produce more nodes in the graph system.

Value

Depending on expand, either a vector or a list with the neighbour nodes to the reference actor(s).

Note

In case that the reference actors are in different components of the network, the output does not discriminate this fact.

Author(s)

Antonio Rivero Ostoic

See Also

expos, rel.sys, bundles

Examples

```r
## Create the data: two binary relations among three elements
arr <- round( replace( array( runif(18), c(3,3,2) ), array( runif(18),
          c(3, 3, 2) ) > .9, 3 ) )

## Determine the system of strong bonds
rs <- rel.sys(arr, bonds = "strong")

## the immediate neighbourhood of the first node
neighb(1, rs)
```

pacnet

Read Output from Pacnet

Description

A function to read output files from the Pacnet program with the full factorization option.

Usage

```r
pacnet(file, toarray = FALSE, uniq = FALSE,
       transp = FALSE, sep)
```
Arguments

file character vector containing a file name or path
toarray (logical) should the induced inclusions be transformed into arrays?
uniq (logical) should only be considered the induced inclusions that are unique?
transp (logical) should the partially ordered structures be transposed?
sep (optional) the pair separator for the pairwise relations

Details

This function is used to read the output file from the Pacnet program, which typically has the .out extension. By default the result is given in a list format, but it is possible to transform the pair lists into arrays. Note that the options in the Pacnet program should include the full factorization in the output; otherwise the object will be NULL.

Value

An object of the 'pacnet' class with items:

ii induced inclusions
at atoms
mc meet complements

Note

Currently only partial order structures of order 36 and less are supported.

Author(s)

Antonio Rivero Ostoic

References


See Also

pi.rels.cngr.decomp.write.dat
Description

Construct the partial order table of unique relations of the semigroup, or else of the concepts produced by Galois derivations.

Usage

```
partial.order(x, type = c("strings", "galois"), lbs, labels)
```

Arguments

- **x**: an object of a 'Strings' or a 'Galois' class
- **type**: whether the object corresponds to strings of relations or Galois derivations
- **lbs**: (optional) the labels of the unique relations
- **labels**: same as lbs for backward compatibility only

Details

To get the partial order of an entire semigroup, both generators and compound relations must be considered. This information and the labels of the unique relations are given by the `strings` function. cf. semigroup to see how the x should be specified properly.

Galois derivations are now possible to be partially ordered as well, and this option is based on the output given by the `galois` function.

Value

An object of 'Partial.Order' class with the partial order table in a matrix form.

Author(s)

Antonio Rivero Ostoic

References


See Also

```
as.strings, strings, galois, perm, diagram, fltr.
```
Examples

```r
## Load the data, and obtain the partial order
data("incubA")

## the strings in the structure
st <- strings(incubA$IM)

## Get the partial order
partial.order(st)
```

---

**perm**  

*Array Permutation*

## Description

Function to permute a given array of relation.

## Usage

```r
perm(x, clu, rev = FALSE, lbs)
```

## Arguments

- `x` a matrix or an array to be permuted
- `clu` the cluster for the permutation
- `rev` (logical) whether the order in `clu` should be reverted.
- `lbs` (optional) the labels after the permutation

## Details

This function serves to permute an array representing relations according to a vector for the clustering membership.

## Value

A permuted matrix or array

## Author(s)

Antonio Rivero Ostoic

## See Also

`cph`, `partial.order`
Examples

```r
## scan the multiplication table data
s <- matrix(data=c(1, 1, 3, 3, 3, 3, 3), nrow=3, ncol=3, byrow=TRUE)

## the permutation as an endomorphism
perm(s, clu = c(1,2,3))
```
Description
A function to preview the partial right multiplication table of the semigroup to assess the size of the complete semigroup.

Usage
prev(x)

Arguments
x an array; usually with three dimensions of stacked matrices where the multiple relations are placed.

Details
When the input data is large, i.e. having a dozen or more elements and/or more than five dimensions, it is recommended to perform this function before the semigroup construction in order to get the partial right multiplication table.

That is because the amount of undefined data in such table gives an idea of how much time may take to get the complete semigroup. However the performance depends mainly on whether the generator matrices are sparse and/or have a relative large number of elements for a semigroup construction of course.

Value
'2stpT' a partial right multiplication table at two-step.
'PcU2stpT' the proportion of undefined elements at two-step.
ordr the dimension of the right multiplication table so far.

Note
The warning message is given only if the percentage of undefined elements and the dimension of the input data are relative high. The semigroup construction can however still take long time without such message; cf. ‘Details’ for this.

Author(s)
Antonio Rivero Ostoic
rbox

See Also

semigroup, edgeT.

Examples

```r
## Create the data: 2 binary relations among 3 elements
arr <- round(replace(array(rnorm(18), c(3, 3, 2)), array(rnorm(18), c(3, 3, 2)) > .5, 1))

## preview it
prev(arr)
```

---

Construct the Relation-Box

Description

Function to construct the Relation-Box of a multiple network

Usage

```r
rbox(w, transp = FALSE, smpl = FALSE, k = 3, tlbs)
```

Arguments

- `w`: an array with three dimensions of stacked matrices of generating relations.
- `transp`: (logical) whether or not the transpose of each matrix in `w` should be included.
- `smpl`: (logical) whether to simplify or not the strings of relations
- `k`: length of the Relation-Box in `z`
- `tlbs`: (optional) a vector with the labels for the transpose relations.

Details

If `transp = TRUE` the labels of the transpose are toggle case of the labels of the original matrices, and in such case it is advised to simplify the strings of relations. In order to prevent a transposed structure for a certain array of `w`, use `NA` in the vector the transpose labels `tlbs` corresponding to the respective matrix.

Value

An object of the ‘Rel.Box’ class.

- `W`: the structure of the Relation-Box
- `lbs`: the labels in the relational system
- `Note`: (optional) Notes indicating the particularities in the input
Orels the original labels of the relations
Srels (optional) the simplified labels of the relations
Trels (optional) the labels of the transposed relations
k the maximal length of the word
z the length of the Relation-Box in the z dimension

Note

Values of k until 9 is supported. With many types of relations, and when the order of the multiple network is high, turning k to more than three may take a long time of computation.

Author(s)

Antonio Rivero Ostoic

References


See Also

cph, semigroup, hierar

Examples

```r
## Create the data: 2 binary relations among 3 elements
c ## the relation box
arr <- round( replace( array(runif(18), c(3,3,2)), array(runif(18), c(3,3,2))>.5, 1 ) )

## The relation box
rbox(arr, k=1)
```

---

**read.dl**

**Read dl Files**

Description

A function to read files with the Ucinet dl format.

Usage

```r
read.dl(file)
```

Arguments

file character vector containing a file name or path of the data representing the network
**Details**

Files d1 serve to represent multiple network structures, and it is one of the formats used in Netdraw, which is a component of the Ucinet program. Besides multiple networks, the function can read two-mode structures as well.

**Value**

a data frame for two-mode networks, or an array representing the multiple networks with one set of actors.

**Note**

The 'EDGELIST' option in DL is not yet supported for reading.

**Author(s)**

Antonio Rivero Ostoic

**References**


**See Also**

`write.dl, read.srt, read.gml`

---

**Description**

A function to read files with the gml format.

**Usage**

```r
read.gml(file, as = c("srt", "array"), directed = TRUE, coords = FALSE)
```

**Arguments**

- `file` character vector containing a file name or path
- `as` should the data be given as a srt or with an array format?
- `directed` (logical) whether the graph is directed or undirected.
- `coords` (logical) whether the coordinates in the gml file should be included.
The gml format, an acronym for \textit{graph modelling language}, provides capabilities to represent multiple networks and add arguments both to the nodes and the edges for visualization purposes.

For the multiplexity in the ties the gml file distinguishes “graphics” arguments inside “edge”. Both “style” and “fill” are supported here and the former has priority over the latter in case the two are given; otherwise when these arguments are absent, the function separates up to a couple of relational levels when several pairwise ties are specified.

Depending the option chosen, the output is either a data frame or an array representing the multigraph. If the coordenates are chosen then these are part of the object structure, but they are not visible.

If the coordenates are chosen, node attributes can also be retrieved.

Antonio Rivero Ostoic

\textit{visone} Software for the analysis and visualization of social networks. \url{http://visone.info}

\textbf{See Also}

\texttt{write.gml, read.srt, read.dl}

\begin{verbatim}
read.srt    Read srt Files
\end{verbatim}

A function to read files with send, receive, and звезд format for a multivariate network with the possibility to transform it into an three dimensional array.

\begin{verbatim}
read.srt(file, header = TRUE, sep = "\t", toarray = TRUE, dichot = FALSE, attr = FALSE, rownames = FALSE, add = NULL)
\end{verbatim}
Arguments

file  path to the file
header (logical) does the file has a header?
sep   the separator among the columns (default is horizontal tab)
toarray (logical) should the data frame be transformed to arrays?
dichot (logical) should the data be dichotomized?
attr  (logical) whether or not the file corresponds to attribute-based data
rownames (logical) are rownames the labels of the nodes?
add   (optional) isolates to be added to the network

Details

srt stands for send, receive, and ties, and it is a data frame with at least 3 columns for the sender, receiver, and the ties, one column for each type of relation. However, the attr option correspond to a actor and self-ties data frame file with the option to transform it into a diagonal matrix. When toarray is set to FALSE, options attr and rownames allow placing the first column of the data frame as the name of the table, which is the format of two-mode data, and compute for instance Galois transformations among the partite sets. If more than one isolate is added, then the data must be included as a vector.

It is also possible to treat the input data as data frame object and manipulate it via e.g. the subset function with the toarray option.

Value

By default an array: usually with three dimensions of stacked matrices where the multiple relations are placed. If toarray = FALSE, then the data frame is given.

Note

Valued networks are now supported as well.

Author(s)

Antonio Rivero Ostoic

See Also

write.srt, read.gml, read.dl, galois
Reduce a matrix or array

**Description**

Function to reduce a matrix or array with a given clustering vector

**Usage**

```
reduc(x, clu, lbs = NULL)
```

**Arguments**

- `x` a matrix or a three-dimensional array to be reduced
- `clu` a vector with the class membership
- `lbs` (optional) the labels to be used in the reduction

**Details**

Given a partition, this function serves to reduce either a matrix representing e.g. a partial order structure. However the reduction is also generalized to three-dimensional arrays representing multiple relations.

**Value**

The reduced matrix or a three-dimensional array.

**Note**

Use `decomp` for the reduction of a semigroup object.

**Author(s)**

Antonio Rivero Ostoic

**See Also**

`cngr, rbox, decomp`

**Examples**

```r
# scan the multiplication table data
s <- matrix(data=c(1, 1, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3), nrow=3, ncol=3, byrow=TRUE)

# Reduce the multiplication table
reduc(s, clu=c(1,2,2))
```
Relational System

Description

Create the Relation System of a multiple network.

Usage

rel.sys(x, type = c("tolist", "toarray"), bonds = c("entire", "strong", "weak", "asym", "recp", "txch", "tent", "mixd", "full"), sel = NULL, loops = FALSE, att = NULL, sep)

Arguments

- **x**: an array; usually with three dimensions of stacked matrices where the multiple relations are placed.
- **type**: if the transformation is from (array of) matrices into lists of pairwise relations or vice versa
- **bonds**: the type of bonds to be used in the creation of the relational system (default the 'entire' network)
- **sel**: (optional) the set of actors to be selected. For "toarray" att and noatt also supported (see details)
- **loops**: (logical) whether or not the loops should be considered in the relational system
- **att**: the arrays in x corresponding to attributes
- **sep**: (optional) the pair separator for the pairwise relations

Details

When the type of bonds chosen is entire then the nodes with ties are considered in the relational system without isolated nodes. Strong bonds are relational bundles with a mutual character, whereas weak bonds are those pattern exclusively without mutual character.

When selecting from a list with actor attributes, it is also possible to select the network members having or not the attribute that is specified in the Attrs output. Use att or noatt for the two options.

Value

An object of 'Rel.System' class for the type = "tolist" (default) option. The items are:

- **ord**: order of the network relational system
- **nodes**: the nodes in the relational system
- **sel**: the selected set of actors
- **sys.ord**: the order of the relational system with the chosen bond type
incl the nodes included the relational system with the chosen bond type
excl the nodes excluded the relational system with the chosen bond type
bond.type the type of bonds used in the relational system creation
size number of ties in the relational system
Note (optional) note
sep the pairwise separator of the relational system
Ties the ties in the relational system
Attrs.ord if att is not NULL, the number of nodes with the chosen attribute(s)
Attrs if att is not NULL, the actors with the chosen attribute(s)

For type = "toarray" the output is a dichotomous 2D or 3D array recording the relations among the actors in the network.

Author(s)
Antonio Rivero Ostoic

References

See Also
expos, bundles, neighb

Examples
## Create the data: two binary relations among three elements
arr <- round(replace(array(runif(18), c(3, 3, 2)), array(runif(18), c(3, 3, 2)) > .9, 3))

## Determine the system of strong bonds
rel.sys(arr, bonds = "strong")

## the first array is for attributes
rel.sys(arr, att = 1)

## select the first node
rel.sys(arr, sel = 1)
R-label

# Relabel a Semigroup Object

Description

Function to relabel the elements of a `Semigroup` class object.

Usage

```r
relabel(S, lbs = NULL)
```

Arguments

- `S`: an object from the `Semigroup` class
- `lbs`: the new labels of the semigroup

Details

Basically this is a function to change the labels of the strings in the multiplication table represented by an object of the `Semigroup` class. Usually this is a consequence of the equations existing in the relational structure.

Value

A `Semigroup` object with the semigroup with the new labels

Note

For the set of equations in the relational structure see the `strings` function.

Author(s)

Antonio Rivero Ostoic

See Also

`semigroup, as.semigroup, convert, strings`

Examples

```r
# Create the multiplication table
s <- matrix(data=c(1, 1, 1, 3, 3, 3, 3, 3, 3), nrow=3, ncol=3, byrow=TRUE)

# make a semigroup object with labels
S <- as.semigroup(s, lbs=c('n', 'm', 'n'))

# convert to a symbolic form a semigroup with relabeled strings
convert(relabel(S, lbs=c('a', 'b', 'c')))
```
rm.isol

Remove Isolates

Description

Function to remove isolate nodes in simple and multiple networks.

Usage

rm.isol(x, diag.incl = TRUE)

Arguments

x a matrix or array representing a network
diag.incl (logical) if arrays, should the diagonals be included?

Details

Isolated nodes do not have any edges in the network, and in a multivariate system, there is no edges adjacent to these kinds of nodes at any level.

Value

The matrix or array representing a multiple network without the isolated actors.

Author(s)

Antonio Rivero Ostoic

See Also

read.srt, zbind

Examples

## Create the data: two binary relations among three elements
arr <- round(replace(array(runif(18), c(3, 3, 2)), array(runif(18), c(3, 3, 2)) > .5, 3))

## Remove isolates (if exist)
rm.isol(arr)
Description

Function to create the complete semigroup of multiple relations, where the multiplication table can
be specified with either a numerical or a symbolic form.

Usage

```r
semigroup(x, type = c("numerical", "symbolic"), lbs = NULL,
        cmp = FALSE, smpl = FALSE)
```

Arguments

- `x`: an array; usually with three dimensions of stacked matrices where the multiple
  relations are placed.
- `type`: does the semigroup should be given in a numerical (default) or in a symbolic
  form?
- `lbs`: (optional) a list of the labels of each distinct relation.
- `cmp`: (optional) a logical to indicate whether the composite matrices should be also
given.
- `smpl`: (logical) whether to simplify or not the strings of relations.

Details

A multiple relation can be defined by square matrices of 0’s and 1’s indicating the presence and
absence of ties among a set of actors. If there is more than one relation type, the matrices must
preserve the label ordering of its elements and stacked into an object array in order to be effectively
applied to this function.

The semigroup, which is an algebraic structure having a set with an associative operation on it, is
calculated considering binary matrices only. This means that if the provided matrices are valued, the
function will dichotomise the input data automatically; values higher or equal to a unit are converted
to one, otherwise they are set to zero. If you are not happy with that, you can go to `dichot` and
specify your own cutoff value for the dichotomization.

Value

An object of ‘Semigroup’ class. The items included are:

- `gens`: an array with the generator relations
- `cmps`: an array with the unique compound relations
- `ord`: a number with the dimension of the semigroup
- `st`: the strings, i.e. a vector of the unique relations
- `S`: the semigroup of relations (see below)
If the specified type is 'numerical', then a matrix of semigroup values is given, otherwise the values is returned as a data frame with the strings of the semigroup.

**Warning**

For medium size or bigger sets (having e.g. more the 4 relation types), the semigroup construction could take a long time.

**Note**

It is recommendable to perform the function `prev` before attempting to construct the semigroup, unless the input data has few dimensions.

**Author(s)**

Antonio Rivero Ostoic

**References**


**See Also**

`prev, strings, edgeT, wordT, relabel, dichot, ltlw, cngr`.

**Examples**

```r
## Create the data: 2 binary relations among 3 elements
arr <- round( replace( array(runif(18), c(3,3,2)), array(runif(18), c(3,3,2))>.5, 1 ) )

## optional: put labels
dimnames(arr)[[3]] <- list("n", "m")

## look at the semigroup
semigroup(arr)
```
semiring

Semiring Structures for Balance Theory

Description

A function to construct semiring structures for the analysis of structural balance theory.

Usage

semiring(x, type = c("balance", "cluster"), symclos = TRUE, transclos = TRUE, k = 2, lbs)

Arguments

x an object of a ‘Signed’ class  
type balance or cluster semiring?  
symclos (logical) apply symmetric closure?  
transclos (logical) apply transitive closure?  
k length of the cycle or the semicycle  
lbs (optional) labels for the semiring output

Details

Semiring structures are based on signed networks, and this function provides the capabilities to handle either the balance semiring or the cluster semiring within the structural balance theory.

A semiring combines two different kinds of operations with a single underlying set, and it can be seen as an abstract semigroup with identity under multiplication and a commutative monoid under addition. Semirings are useful to determine whether a given signed network is balanced or clusterable. The symmetric closure evaluates this by looking at semicycles in the system; otherwise the evaluation is through closed paths.

Value

An object of ‘Semiring’ class. The items included are:

val the valences in the semiring  
s the original semiring structure  
Q the resulted semiring structure  
k the number of cycles or semicycles

Note

Disabling transitive closure should be made with good substantial reasons.
**Author(s)**

Antonio Rivero Ostoic

**References**


**See Also**

`signed`, `as.signed`

**Examples**

```r
# Create the data: two sets with a pair of binary relations
# among three elements
arr <- round( replace( array( runif(18), c(3,3,2) ), array( runif(18), c(3,3,2) ) > .5, 3 ) )

# Make the signed matrix with two types of relations
sg <- signed(arr)

# Establish the semiring structure
semiring(sg)
```

**Description**

Construct the signed network of a system of contrasting relations

**Usage**

`signed(P, N = NULL, lbs)`

**Arguments**

- `P` array with the positive ties and possible with negative ties (see Details)
- `N` (optional) array with the negative ties
- `lbs` (optional) labels for the signed matrix
**Details**

This function coerce a array(s) to become a ‘Signed’ object. Positive ties are always in the first argument, and in case that this array has three dimensions, then the second dimension is considered as the negative ties, provided that N is NULL. If ambivalent ties are present in the structure then the signed matrix represent positive, negative, ambivalent, and null ties as p, n, a, and o respectively; otherwise the values are 1, -1, and 0.

**Value**

An object of ‘Signed’ class with items:

- `val` the valences in the signed matrix
- `s` the signed matrix

**Note**

A warning message is shown when the N argument has more than two dimensions.

**Author(s)**

Antonio Rivero Ostoic

**References**


**See Also**

`semiring, as.signed`

**Examples**

```r
## Load the data
data("incubA")

## Make the signed matrix with two types of relations
signed(incubA$IM)
```
Strings of Relations

Description

Function to get the labels of the unique relations of the semigroup; that is the generators and compound relations that are the elements of the complete semigroup.

Usage

strings(x, smpl = FALSE, equat = FALSE, k = 2)

Arguments

x

an array; usually with three dimensions of stacked matrices where the multiple relations are placed.

smpl

(logical) whether to simplify or not the strings of relations

equat

(logical) should the equations be included in the output?

k

length of the strings

Details

The strings are the unique relations which constitutes the elements of the complete semigroup. These are both the generators and the compound relations after applying the Axiom of Quality, which means that even some generators can be disregarded.

This function is especially useful to construct the partial order of relations, and to establish the set of equations in the relational structure.

Value

An object of ‘Strings’ class.

wt

the generators and compound relations

ord

the order of the structure

st

the labels of the unique relations

equat

the equations among strings of relations

Note

The maximum length of the strings in the equations is currently 3.

Author(s)

Antonio Rivero Ostoic
**summaryBundles**

**References**


**See Also**

`partial.order, semigroup, ltlw`.

**Examples**

```r
## Create the data: 2 binary relations among 3 elements
arr <- round(replace(array(runif(18), c(3, 3, 2)),
                     array(runif(18), c(3, 3, 2)) > .5, 1))

## get the strings
strings(arr)
```

---

| summaryBundles | Summary of Bundle Classes |

**Description**

Pretty printing of the bundle class patterns results.

**Usage**

`summaryBundles(x, file = NULL, latex = FALSE, byties)`

**Arguments**

- `x`: an object of the ‘Rel.Bundles’ class
- `file`: (optional) the path where the output file is to be placed
- `latex`: (logical) whether the output should be in latex format or not
- `byties`: (optional and logical) expand tie patterns and collapse tie labels?

**Details**

This function prints the bundle census patterns existing in the network with an option to export such information in a friendly format. The dyadic bundle patterns are provide by the function `bundles`; however, the outcome of this function provides a list of pair lists for each bundle with the involved types of relations and nodes in the network. This form for presentation, although is convenient for further computation, it is not always easy to read for a human eye. The pair separator used to print the bundle occurrences is taken from the output of the `bundles` function.

If `latex` is set to `TRUE`, then the path file is activated to obtain a `tex` file with the different bundle class patterns. Finally, the optional argument `byties` provides more precise information of the patterned ties disregarding the relational content.
Value

The distinct bundle class patterns with a user friendly format.

Note

In case that the file already exists in the pointed directory, then the file will be overwritten.

Author(s)

Antonio Rivero Ostoic

References


See Also

`bundles, bundle.census`

Examples

```r
# Create the data: 2 binary relations among 3 elements
arr <- round( replace( array( runif(18), c(3 ,3, 2) ), array( runif(18), c(3, 3, 2) ) > .8, 3 ) )

# Establish the different bundles
bd <- bundles(arr)

# Print the different relational bundles
summaryBundles(bd)
```

---

**transf**

Transform Data from/to Matrix/List Formats

Description

Function to transform data from/to matrix/list formats representing a network.

Usage

```r
transf(x, type = c("tolist", "toarray"), lbs = NULL, lb2lb, sep, ord)
```
Arguments

- **x**: an array or a list of pair relations
- **type**: whether the transformation is from a matrix to a list of pair relations, or from a list of pair relations to an array format
- **lbs**: (optional) the labels in the transformation
- **lb2lb**: (optional and logical) whether the transformation is label-to-label. Default TRUE for "toarray" and FALSE for "tolist"
- **sep**: (optional) the pair separator for the pairwise relations
- **ord**: (optional) the order of the resulted structure ('toarray' option, otherwise ignored)

Details

'tolist' is the option to transform a matrix or an array to a list of pair elements. In case that the lb2lb is enabled in this type of transformation, then lbs must be provided, whereas the pair separator is optional. On the other hand 'toarray' will produce a matrix from a list of pair elements, and in this case is advisable to specify the order of the structure. Three dimensional structures are now supported.

Data frames are also accepted for the 'tolist' option, but in case that this information is given as a list of pair relations the output will be a square matrix.

Value

Depending on the input data, the result is either a list of pair relations or a matrix of relations.

Note

For high dimensional arrays, the `rel.sys` function provides additional information other than the list of pair relations of the entire structure.

Author(s)

Antonio Rivero Ostoic

See Also

- `read.srt`, `bundles`, `reduc`, `rel.sys`

Examples

```r
## scan the multiplication table data
s <- matrix(data=c(1, 1, 1, 3, 3, 3, 3, 3, 3), nrow=3, ncol=3, byrow=TRUE)

## transform the matrix to a list format
transf(s, lb2lb = TRUE, lbs = c('n','m','n'))
```
The Word Table of Relations

Description
The Word Table of multiple relations.

Usage
wordT(x)

Arguments
x an array; usually with three dimensions of stacked matrices where the multiple relations are placed.

Details
The Word Table is a consequence of the Edge Table and the function gives a list of indexed elements in the complete semigroup.

In terms of the Cayley graph of the semigroup, the collection of unique relations (both compound an generators) are represented by nodes, and the generators are edges that record the result of post-multiplying the compound relations by the generators (Pattison, 1993).

Value
gens the generator relations
\( WT \) the Word Table where "n" stands for "node" and "g" stands for "generator"

The generators do not have values neither in the "node" nor the "generator" of the Word table since they are not product of any other element in the semigroup. cf. details for the rest of the values.

Note
The labels for the elements can be retrieved by the strings function.

Author(s)
Antonio Rivero Ostoic

References
See Also

edgeT, semigroup.

Examples

## Create the data: 2 binary relations among 3 elements
arr <- round(replace( array(runif(18), c(3,3,2)), array(runif(18), c(3,3,2))>.5, 1 ))

## get the word table
wordT(arr)

write.dat Files

Description

A function to write dat files.

Usage

write.dat(x, path)

Arguments

x an object representing the multiple network structure
path the path file for the output

Details

‘dat’ files are the format used in the Pacnet program. In case that the input data represents a multiple network then a separate file will be produced, each one representing a single type of relationship in the system. The name of the output files depends on the object title.

Value

File(s) with adjacency matrices with a .dat format

Note

In case that the directory in the path for the output does not exist then it will be created automatically.

Author(s)

Antonio Rivero Ostoic
References

StOCNET An open software system for the advanced statistical analysis of social networks.
http://www.gmw.rug.nl/~stocnet/

See Also

pacnet, write.gml, write.dl

write.dl | Write dl Files

Description

A function to write dl files representing multiple networks.

Usage

write.dl(x, file = NULL, type = c("nodelist", "fullmat"))

Arguments

x an object representing the multiple network
file path to the file
type whether to write the data as a nodelist or as a fullmat format

Details

dl files serve to represent multiple networks, and it is one of the formats used in Netdraw, which is a component of the UCINET program.

Value

A file with the data with a .dl format

Author(s)

Antonio Rivero Ostoic

References


See Also

read.dl, write.gml, write.srt, write.dat
Description

A function to write files with a gml format.

Usage

write.gml(x, file = NULL)

Arguments

x an object representing the multiple network
file path to the file

Details

The gml format, an acronym for graph modelling language, provides capabilities to represent multiple networks and add arguments both to the nodes and the edges for visualization purposes.

Value

A file with the data with a graph modelling language format.

Note

In case that the file already exists in the pointed directory, then the file will be overwritten.

Author(s)

Antonio Rivero Ostoic

References


See Also

read.gml, write.dl, write.dat
write.srt

Write .srt Files

Description

A function to write .srt files

Usage

write.srt(x, file = NULL, sep = "\t", header = TRUE)

Arguments

x an object representing the multiple network
file path to the file
sep the separator used between the columns
header (logical) whether the header should be included in the file

Details

.srt stands for send, receive, and ties, and it is a data frame with at least 3 columns for the sender, receiver, and the ties, one column for each type of relation.

Value

A file with the data with a .srt format

Author(s)

Antonio Rivero Ostoic

See Also

read.srt, write.dl
**Combine Arrays**

**Description**

Combine multidimensional arrays.

**Usage**

```r
zbind(...)```

**Arguments**

...  

One or more arrays with two or three dimensions

**Details**

This function represents an extension to both the `cbind` and the `rbind` functions, and it is especially useful in constructing a single object to represent a multivariate system structure. Both square and rectangular arrays are supported provided that the dimensions in the input are equal.

**Value**

Usually a three dimensional array

**Note**

Data frames should be transformed into arrays

**Author(s)**

Antonio Rivero Ostoic

**See Also**

`mnplx`, `dichot`, `strings`

**Examples**

```r
## Create the data: two sets with a pair of binary relations  
## among three elements
arr1 <- round( replace( array( runif(18), c(3,3,2) ), array( runif(18),  
                c(3, 3, 2) ) > .5, 3 ) )

arr2 <- round( replace( array( runif(18), c(3,3,2) ), array( runif(18),  
                c(3, 3, 2) ) > .5, 3 ) )

## bind the data sets
zbind(arr1, arr2)```
Index

*Topic IO
  - multiplex-package, 3
  - pacnet, 33
  - read.dl, 40
  - read.gml, 41
  - read.srt, 42
  - summaryBundles, 55
  - write.dat, 59
  - write.dl, 60
  - write.gml, 61
  - write.srt, 62

*Topic algebra
  - as.semiGroup, 5
  - as.strings, 7
  - cngr, 11
  - cph, 14
  - decomp, 15
  - diagram.levels, 18
  - edgeT, 20
  - fltr, 23
  - galois, 24
  - hierar, 25
  - iinc, 26
  - is.mc, 28
  - ltlw, 30
  - partial.order, 35
  - perm, 36
  - pi.rels, 37
  - prev, 38
  - rbox, 39
  - semigroup, 49
  - semiring, 51
  - strings, 54
  - wordT, 58

*Topic array
  - as.signed, 6
  - as.strings, 7
  - convert, 13
  - cph, 14
  - decomp, 15
  - mnplex, 31
  - perm, 36
  - reduc, 39
  - rm.isol, 44
  - signed, 52
  - strings, 54
  - transf, 56
  - zbind, 63

*Topic attribute
  - bundles, 9
  - expos, 21
  - multiplex-package, 3
  - rel.sys, 45

*Topic cluster
  - cngr, 11
  - comps, 12
  - decomp, 15
  - perm, 36
  - reduc, 44

*Topic datasets
  - incubs, 27

*Topic data
  - bundle.census, 8
  - bundles, 9
  - multiplex-package, 3
  - pacnet, 33
  - read.dl, 40
  - read.gml, 41
  - read.srt, 42
  - write.dat, 59
  - write.dl, 60
  - write.gml, 61
  - write.srt, 62

*Topic file
  - multiplex-package, 3
  - pacnet, 33
  - read.dl, 40
read.gml, 41
write.dat, 59
write.dl, 60
write.gml, 61
write.srt, 62
*Topic graphs
  diagram, 17
*Topic list
  bundles, 9
*Topic manip
  convert, 13
  diagram.levels, 18
  dichot, 19
  iinc, 26
  isom, 29
  mnplx, 31
  multiplex-package, 3
  neighb, 32
  perm, 36
  read.srt, 42
  reduc, 44
  relabel, 47
  rm.isol, 48
  transf, 56
  zbind, 63
*Topic math
  as.semigroup, 5
  bundles, 9
  cngr, 11
  cph, 14
  decomp, 15
  fltr, 23
  galois, 24
  hierar, 25
  ltlw, 30
  partial.order, 35
  perm, 36
  prev, 38
  semigroup, 49
  semiring, 51
*Topic models
  as.signed, 6
  comps, 12
  expos, 21
  multiplex-package, 3
  neighb, 32
  rel.sys, 45
  semiring, 51
  signed, 52
*Topic print
  summaryBundles, 55
as.semigroup, 5, 14, 47
as.signed, 6, 52, 53
as.strings, 7, 18, 35
bmgraph, 4
bundle.census, 8, 10, 56
bundles, 8, 9, 13, 22, 33, 46, 55-57
cbind, 63
cngr, 11, 16, 29, 30, 34, 44, 50
comps, 12
convert, 5, 6, 13, 47
cph, 14, 26, 27, 36, 40
decomp, 12, 15, 29, 34, 38, 44
diagram, 15, 17, 18, 19, 24-26, 35
diagram.levels, 18, 18
dichot, 19, 22, 32, 49, 50, 63
dgelT, 20, 39, 50, 59
expos, 21, 33, 46
fltr, 23, 25, 35
galois, 18, 24, 24, 35, 43
hierar, 25, 40
iinc, 26, 29
incuba (incubs), 27
incubB (incubs), 27
incubC (incubs), 27
incubs, 27
is.mc, 28
isom, 29
ltlw, 30, 50, 55
mnplx, 31, 63
multigraph, 4
multiplex-package, 3
neighb, 22, 32, 46
pacnet, 12, 27, 33, 38, 60
partial.order, 7, 16–19, 24–27, 35, 36, 55
perm, 19, 35, 36
pi.rels, 16, 29, 34, 37
prev, 20, 38, 50

rbind, 63
rbox, 15, 26, 39, 44
read.dl, 40, 42, 43, 60
read.gml, 41, 41, 43, 61
read.srt, 41, 42, 48, 57, 62
reduc, 16, 30, 32, 44, 57
rel.sys, 13, 22, 33, 45, 57
relabel, 14, 47, 50
replace, 20
rm.isol, 48

semigroup, 5, 6, 14–16, 20, 21, 27, 30, 31, 35,
39, 40, 47, 49, 55, 59
semiring, 6, 51, 53
signed, 6, 52, 52
strings, 7, 18, 31, 35, 47, 50, 54, 58, 63
subset, 43
summaryBundles, 9, 10, 55

transf, 10, 56

wordT, 21, 50, 58
write.dat, 34, 59, 60, 61
write.dl, 41, 60, 60, 61, 62
write.gml, 42, 60, 61
write.srt, 43, 60, 62

zbind, 7, 32, 48, 63